비트캠프 자바
오락실 콘솔게임팀

팀원(가나다순): 강다희, 박수현, 손민희, 임욱표, 정건, 한혜원

목차

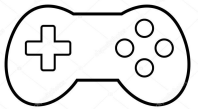
1. 프로젝트에 대한 설명&시연

2. 각 기능별 설명(구현기능, 발생문제 및 해결방법)

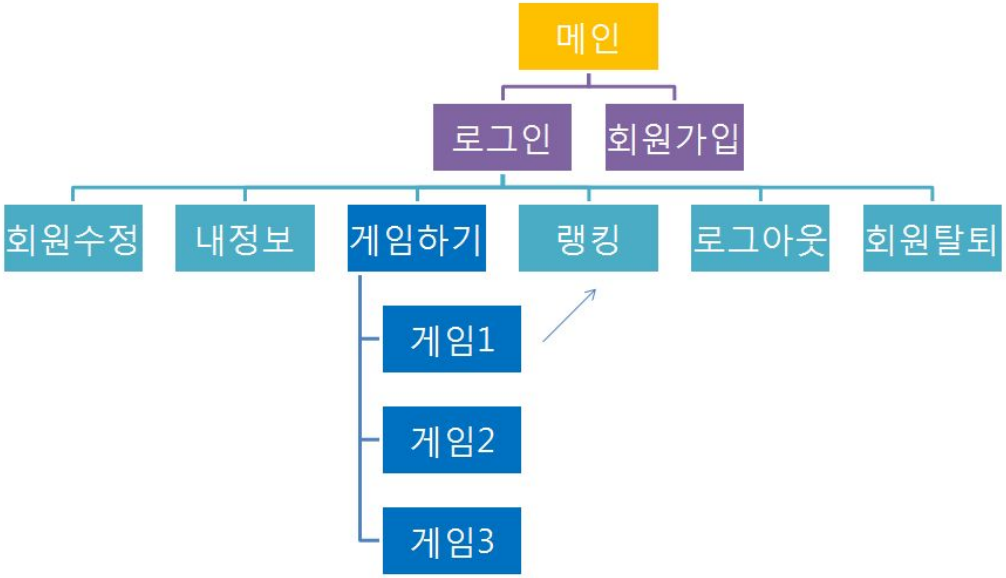
- 메인 및 회원관리
- 게임1: 억울한 사형수를 살려라!
- 게임2: 영화제목을 맞춰라!
- 게임3: 가위바위보!
- 랭킹관리

3. 협업 과정에서의 문제점

4. 프로젝트의 개선방향



1. 프로젝트에 대한 설명 & 시연



메인 및 회원관리: 강다희

게임 1(행맨): 손민희(팀장)

게임 2(영화이름): 임욱표

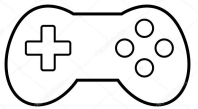
게임 3(가위바위보): 한혜원

게임 랭킹: 정건

2. 메인/회원관리

1. 전체구조 및 구현기능

- `public class UserInfo {}` //정보저장목적
- `public interface UserInfoInterface {}` //필요한메서드정의
- `public class UserMain {}` //메인클래스
- `public class UserManager{}` //인터페이스상속하는 기능클래스



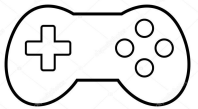
2. 메인/회원관리

1. 전체구조 및 구현기능

- `public class UserInfo {`

```
private String id;        //아이디
private String password;//비밀번호
private int score;        //점수

//점수는 0으로 초기화
public UserInfo(String id, String pw) {
            this(id, pw, 0);
      }
}
```

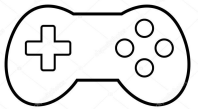
```
public String getId() {
        return id;
}

public void setId(String id) {
        this.id = id;
}
```



2. 메인/회원관리

1. 전체구조 및 구현기능

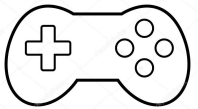
- `public class UserInfoManager implements UserInfoInterface{`
  - `public int printMainMenu()` //메인메뉴 출력
  - `public void mainMenu()` //메뉴번호를 받아와 메서드 호출
  - `public void login()` //로그인 메서드
  - `public void joinMember()` //회원가입 메서드
  - `public void subMain(String id)` //로그인 후 이동되는 서브메인메뉴메서드



2. 메인/회원관리

1. 전체구조 및 구현기능

```
- public void updateInfo(String id)      //비밀번호변경 메서드
- public void myInfo(String id)          //내정보보기 메서드
- public void gameMain(String id)        //게임선택 메뉴 메서드
- public void rank()                     //랭킹출력 메서드
- public void deleteMember(String id)    //회원탈퇴메서드

}
```



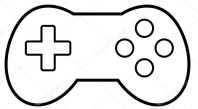
2. 메인/회원관리

1. 전체구조 및 구현기능

```
public static Map<String, UserInfo> userinfo = new HashMap<String, UserInfo>();
```

- `public void joinMember(){`

```
// id가 중복이아니라면 break를하여 비밀번호를 받으러감
if (!(userinfo.containsKey(id))) {
        break;
} else {
        System.out.println("중복된 아이디입니다. 다시 입력해주세요.");
}

//비밀번호까지 받은 후
        UserInfo ui = new UserInfo(id, pw);
        userinfo.put(ui.getId(), ui);
                              key값  ,  value값
}
```



2. 메인/회원관리

1. 전체구조 및 구현기능

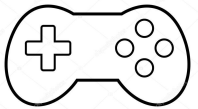
- `public void login(){`

```
if (!(userinfo.containsKey(id))) {

        System.out.println("회원정보가 없습니다. \n회원가입해주세요.");

} else if (userinfo.containsKey(id) && userinfo.get(id).getPassword().equals(pw)) {
        System.out.println("로그인되었습니다.");
        System.out.println(id + "님 환영합니다.");

        subMain(id);
} else {
        System.out.println("아이디와 비밀번호를 확인해주세요.");
        login();
}
```

userinfo에 같은 id를 가진 유저가 있는지 확인

userinfo에 등록된 id와 pw가 같으면 로그인해줌

로그인 후 들어갈 수 있는 메뉴로 이동시켜줌 (id값을 넘겨서 로그인상태를 유지)



2. 메인/회원관리

1. 전체구조 및 구현기능

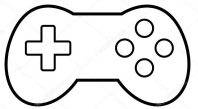
- `public void printRank(String id){`

  id 정보를 매개변수로 전달하여 로그인을 유지할 수 있도록 함

  `lankTest.RankTest.rank(id);`

  패키지명.클래스명.메서드명() 으로 가져와서 연결

  `}`

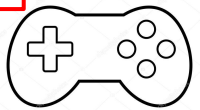


2. 메인/회원관리

2. 구현 중 발생했던 문제 & 해결방법

```
// 서브메뉴출력
public void printSubMenu(String id, String nName) {

    System.out.println("====================================");
    System.out.println("          Sub Menu");
    System.out.println("====================================");
    System.out.println("로그인 유저 : " + id + "(" + userinfo.get(id).getNickname
    System.out.printf("%d.정보수정\n%d.내정보\n%d.게임하기\n%d.랭킹\n%d.로그아웃\n%d
            Menu.L_RANK, Menu.L_OUT, Menu.L_WITHDRAWAL);
    System.out.println("\n====================================");

    // 숫자가아닌 문자가 들어왔을 때
    checkInt();


    int subChoice = Util.keyboard.nextInt();
    Util.keyboard.nextLine();// 현재 라인의 버퍼를 출력(clear)

    subMain(subChoice, id);

}


//서브메뉴관리
public void subMain(int subChoice, String id) {

    switch (subChoice) {
    case Menu.L_UPDATE://정보수정
        System.out.println("====================================");
        System.out.println("          정보수정");
        System.out.println("====================================");
        updateInfo(id);
        break;
    case Menu.L_INFO://내정보
        System.out.println("====================================");
        System.out.println("          내정보 보기");
        System.out.println("====================================");
```

```
// 서브메뉴관리
public static void subMain(String id) {

    while (true) {

        System.out.println("====================================");
        System.out.println("          Sub Menu");
        System.out.println("====================================");
        System.out.println("로그인 유저 : " + id);
        System.out.printf("%d.내정보확인\n%d.비밀번호변경\n%d.게임하기\n%d.랭킹
                Menu.L_RANK, Menu.L_OUT, Menu.L_WITHDRAWAL);
        System.out.println("\n====================================");

        int subChoice = Util.keyboard.nextInt();
        Util.keyboard.nextLine();// 현재 라인의 버퍼를 출력(clear)
//      checkInt();
        switch (subChoice) {
        case Menu.L_UPDATE:// 정보수정
            System.out.println("====================================");
            System.out.println("          비밀번호 변경");
            System.out.println("====================================");
            updateInfo(id);
            break;
        case Menu.L_INFO:// 내정보
            System.out.println("====================================");
            System.out.println("          내정보 보기");
            System.out.println("====================================");
            myInfo(id);
            break;
```

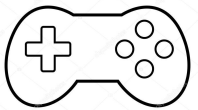


2. 메인/회원관리

2. 구현 중 발생했던 문제 & 해결방법

```
switch (subChoice) {
case Menu.L_UPDATE:// 정보수정
    System.out.println("=====================================");
    System.out.println("          비밀번호 변경");
    System.out.println("=====================================");
    updateInfo(id);
    break;
case Menu.L_INFO:// 내정보
    System.out.println("=====================================");
    System.out.println("          내정보 보기");
    System.out.println("=====================================");
    myInfo(id);
    break;
case Menu.L_GAME:// 게임하기
    gameMain(id);
    break;
case Menu.L_RANK:// 랭킹
    System.out.println("=====================================");
    System.out.println("          랭킹확인");
    System.out.println("=====================================");
    printRank(id);
    break;
case Menu.L_OUT:// 로그아웃
    System.out.println("로그아웃됩니다.");
    mainMenu();
    break;
case Menu.L_WITHDRAWAL:// 회원탈퇴
    deleteMember(id);
    break;
default:
    System.out.println("메뉴중에서 선택해주세요.");
    break;
}
```

```
    break;
case Menu.L_OUT:// 로그아웃
    System.out.println("로그아웃됩니다.");
    //mainMenu();
    break;
case Menu.L_WITHDRAWAL:// 회원탈퇴
    deleteMember(id);
    break;
default:
    System.out.println("메뉴중에서 선택해주세요.");
    break;
}
if(subChoice == Menu.L_OUT) {
    break;
}
```



3. 게임1: 억울한 사형수를 살려라!(a.k.a. 행맨게임)

1. 전체구조 및 구현기능

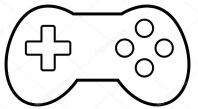
- 게임규칙
  - 일정한 글자수의 단어 추측
  - 정답이면 → 빈칸이 추측 알파벳으로
  - 오답이면 → 행맨의 몸이 하나씩 지워짐
  - 유저는 알파벳과 단어 두 개의

    방법으로 추측할 수 있다.

3. 게임1: 억울한 사형수를 살려라!(a.k.a. 행맨게임)

1. 전체구조 및 구현기능

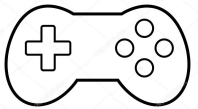
- `class GameFlow extends UserManager {} //게임전체흐름`
- 게임의 난이도 선택: switch 문 → 초급, 중급, 고급
- 뒤로가기: 바로 전 단계로 이동



3. 게임1: 억울한 사형수를 살려라!(a.k.a. 행맨게임)

1. 전체구조 및 구현기능

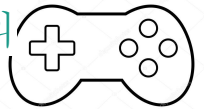
- `class RandomWords {} //정답단어 랜덤뽑기`
- shuffle 매서드 사용을 위해 ArrayList 저장방법 이용
- `ArrayList<String> gameWord1 = new ArrayList<String>();`
- `gameWord1.add("ant");`
- `Collections.shuffle(gameWord1);`



3. 게임1: 억울한 사형수를 살려라!(a.k.a. 행맨게임)

1. 전체구조 및 구현기능

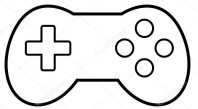
- public class **GameManager** extends UserManager {
  - public int **startMenu**(String id) {} //게임1 메뉴시작
  - void **gameStart**(int n) {} //게임1 시작
  - void **afterGame**(int failCnt) {} //게임 완료 후 메뉴창
  - String **deleteHangMan**(int n) {} //행맨 지우기
  - void **gamePoint**(int n) {} //게임 포인트 산출
  - public void **savePoint**(int point) {} } //랭킹의 게임포인트 저장 매서드 오버라이딩



3. 게임1: 억울한 사형수를 살려라!(a.k.a. 행맨게임)

1. 전체구조 및 구현기능

- `public class GameManager extends UserManager {`
  - `void gameStart(int n) {} }` `//게임1 시작`
  - `String correctAnswer`
  - `char[] correctAnswerArray = correctAnswer.toCharArray();`
  - `char[] blank = new char[correctAnswerArray.length];`
  - `for (int i = 0; i < correctAnswerArray.length; i++)`
    `{blank[i] = '_'; }`
  - `int failCnt = 0; // 유저의 오답 횟수`

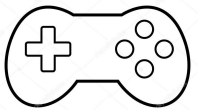


3. 게임1: 억울한 사형수를 살려라!(a.k.a. 행맨게임)

1. 전체구조 및 구현기능

- `public class GameManager extends UserManager {`
  - `void gameStart(int n) {} }` `//게임1 시작`
  - `while(12-failCnt) {`
    - 경우1: 유저가 알파벳을 입력할때
      - 정답이면: '_' → '알파벳'
      - 오답이면: 핵맨의 신체부위 사라짐
    - 경우2: 유저가 단어를 입력할때
      - 정답이면: 천재이군요! → 게임 포인트 저장/게임 이후 메뉴선택
      - 오답이면: 핵맨의 신체부위 사라짐

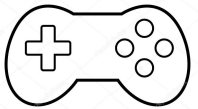
  `}`



3. 게임1: 억울한 사형수를 살려라!(a.k.a. 행맨게임)

1. 전체구조 및 구현기능

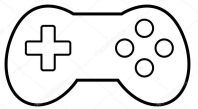
- `public class GameManager extends UserManager {`
  - `void afterGame(int failCnt) {}  //게임 완료 후 메뉴창`
    - 1번: 다시 게임하기
    - 2번: 메인으로 돌아가기
  - `public void savePoint(int point) {} }`
    - 랭킹의 GameBoy 클래스의 savePoint()를 오버라이딩



3. 게임1: 억울한 사형수를 살려라!(a.k.a. 행맨게임)

2.구현 중 발생했던 문제 & 해결방법

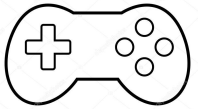
- 정답단어배열과 빈칸배열의 비교 → 전체 정답 단어와의 일치 확인
- `if (correctAnswer.equals(String.valueOf(blank))) {`
- `System.out.println("당신은 천재이군요!"); break; }`



4. 게임2: 영화제목을 맞춰라!

1. 전체 구조 및 구현기능

- `public class Game implements GameInterface{}`
- `public class GameMain {}`
- `public class LevelOne extends Game {}`
- `public class LevelTwo extends Game {}`
- `public class LevelThree extends Game{}`
- `public class Util{}`
- `public class Time extends Game{}`
- `public class GameInterface{}`

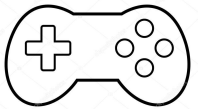


4. 게임2: 영화제목을 맞춰라!

2. 실행 흐름

- public class GameMain {}

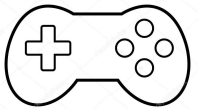
1. 영화 장르 선택(switch)
2. 난이도 선택(switch)
3. 문제 풀이(if 문 사용으로, 정답을 맞췄을 경우->저장 후 메인, 틀렸을 경우 메인으로 이동)
4. 메인으로 이동



4. 게임2: 영화제목을 맞춰라!

2. 구현 중 발생했던 문제 & 해결방법

1. 문제를 풀지 못했을 때 타이머 쓰레드가 종료되고 바로 메인으로 넘어가는 구조로 만들고 싶었으나, 타이머가 끝난 뒤 입력을 해야 다음으로 넘어가는 문제가 발생 - 입력 값을 주었음.
2. 문제를 빨리 풀고 다음 문제로 넘어가면 타이머 쓰레드가 중첩 되는 문제가 발생 - 타이머 종료를 사용하여 정답을 맞췄을 경우 타이머가 계속해서 진행되지 않지만 중첩 오류는 막을 수 없었음.



4. 게임2: 영화제목을 맞춰라!

3. 문제점 : 코드의 중복

```
// 한국영화 문제1
public void Korea1() {
    System.out.println("난이도 초급 문제입니다.");
    System.out.println("[ㅇㅇㅇ ㅇ도]");
    answer2();// 출력메소드

    time.Timer();// 시간 초 시작

    while (time.count < 60) {

        String answer = util.key.nextLine();

        if (answer.equals("연애의 온도")) {
            time.cancel();// 시간 초 종료
            saveData();// 저장 메소드
            answer();// 정답시 출력 메소드
            break;
        } else if (answer.equals("힌트")) {
            System.out.println("<힌트 : 출연배우 이민기,
        } else if (answer.equals("난 바보야")) {
            break;
        } else {
            WrongAnswer();// 틀렸을시 출력 메소드
```

```
// 애니영화 문제1
public void ani2() {
    System.out.println("난이도 중급 문제입니다.");
    System.out.println("[ㅇㅅㅇㄷ ㅇㅇ]");
    answer2();//출력메소드
    time.Timer();

    while (time.count < 60) {
        String answer = util.key.nextLine();
        if (answer.equals("인사이드 아웃")) {
            time.cancel();
            answer();//정답메소드
            saveData();//저장메소드
            break;
        } else if (answer.equals("힌트")) {
            System.out.println("<힌트 : 감정>");
        } else if (answer.equals("난 바보야")) {
            break;
        } else {
            WrongAnswer();//틀렸을때 출력 메소드
```

```
// 애니영화 문제3
public void ani3() {
    System.out.println("난이도 고급 문제입니다.");
    System.out.println("[ㅇㅇㅅㅇㅇㅈ]");
    answer2();//출력메소드
    time.Timer();

    while (time.count < 60) {

        String answer = util.key.nextLine();

        if (answer.equals("아이스에이지")) {
            time.cancel();
            answer();//정답 메소드
            saveData();
            main();
            break;
        } else if (answer.equals("힌트")) {
            System.out.println("<힌트 : 도토리>");
        } else if (answer.equals("난 바보야")) {
            main();
        } else {
            WrongAnswer();//틀렸을때 출력 메소드
```

5. 게임3: 가위바위보! 혜원이바보멍청이

1. 전체 구조 및 구현기능
   - `public class Server {}`
   - `public class Client {}`
   - `public class EchoThread extends Thread {}`
   - `public class Game {}`

5. 게임3: 가위바위보!

1. 전체 구조 및 구현기능

- public class Server {}

```
//  서버소켓 생성
        serverSocket = new ServerSocket(7777);
//  클라이언트의 접속요청을 대기한다.
        System.out.println("서버가 열렸습니다.");
        socket = serverSocket.accept();
        System.out.println("클라이언트가 접속했습니다.");
//자료를 송수신하게 해주는 별도의 쓰레드
EchoThread echoThread = new EchoThread(socket);
//에코 쓰레드 실행
```

5. 게임3: 가위바위보!

1. 전체 구조 및 구현기능

- public class Client {}

```
socket = new Socket("10.10.10.28", 7777);//서버와 연결
InputStream in = socket.getInputStream(); // 수신 --> read();
BufferedReader br = new BufferedReader(new InputStreamReader(in));
OutputStream out = socket.getOutputStream();// 송신 --> write();
PrintWriter pw = new PrintWriter(out);

BufferedReader stdin = new BufferedReader(new
InputStreamReader(System.in));
```

5. 게임3: 가위바위보!

1. 전체 구조 및 구현기능

- public class Client {}

```
//첫 화면 출력
Game game = new Game();
int cnt = 1;
str = stdin.readLine();
cnt = game.getCount(); // 키보드로부터 데이터 입력받기
if(cnt<9)//숫자입력
else if(cnt == 10)//숫자입력, 점수 출력, break
```

6. 게임랭킹관리

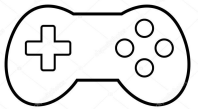
전체구조 및 구현기능

```
class Game

   public void saveData(int check)

   void dataUp(List<GameBoy> list)
```

```
class Comprator
```

```
class Rank
      ↓
class RankManger
```

```
   void checkRank(List<GameBoy> list)

   System.out.println("==RANKING CHECK==");
   System.out.println("1.FirstGame\n2.SecondGame\n3.ThirdGame\n4.Fourth Game\n5.EXIT RANK");
                                  ↓
   System.out.println("1.ALL RANK\n2.MY RANK\n3.BACK");
```

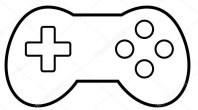


6. 게임랭킹관리

1. 배열과 리스트

a. 배열일경우 - > 인원수 제한
b. 리스트일경우- > 점수순으로 정렬 - > sort(), comparator

```
//소트정렬을 위한 클래스
class Comprator implements Comparator<GameBoy> {
    @Override
    public int compare(GameBoy o1, GameBoy o2) {
        return o2.point - o1.point;
    }
}

Collections.sort(list, new Comprator());
```



6. 게임랭킹관리

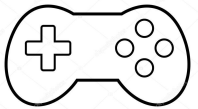
2. 저장

remove()로 지우고 add로추가? - - > **X**

->set()

```
public void saveData(int check) {
    // 게임 선택에서 고른수를 가져와 그 게임에 맞는 결과를 저장한다

    switch (check) {
    case 1:
         lt.list = lt.rm.lank1;
        dataUp(lt.list);

        break;
    case 2:
        lt.list = lt.rm.lank2;
        dataUp(lt.list);
        break;
    case 3:
        lt.list = lt.rm.lank3;
        dataUp(lt.list);
        break;
    case 4:
        lt.list =lt. rm.lank4;
        dataUp(lt.list);
        break;

    }
```

```
//점수갱신
    void dataUp(List<GameBoy> list) {
        int index = -1;
        Collections.sort(list, new Comprator());
        for (int i = 0; i < list.size(); i++) {
            if (name.equals(list.get(i).name) && point >= list.get(i).point) {
                index = i;
                break;
            }
        }
        if (index < 0) {
            System.out.println("정보가 저장되었습니다");
            list.add(new GameBoy(name, point));
            showData();
        } else {
            System.out.println("새로운 점수로 갱신되었습니다");
            list.set(index, new GameBoy(name, point));
            showData();
        }
    }
}
```



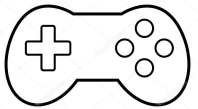
6. 게임랭킹관리

3. 코드의 반복

```
List<GameBoy> lank1 = new ArrayList<GameBoy>();
List<GameBoy> lank2 = new ArrayList<GameBoy>();
List<GameBoy> lank3 = new ArrayList<GameBoy>();
List<GameBoy> lank4 = new ArrayList<GameBoy>();
```

→

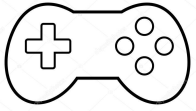
```
static List<GameBoy> list;
```



6. 게임랭킹관리

3. 코드의 반복 - 전

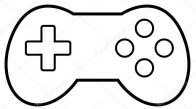
```
package lankTest;


   import java.util.ArrayList;

   public class BeforeLankManager {


       List<GameBoy> lank1 = new ArrayList<GameBoy>();
       List<GameBoy> lank2 = new ArrayList<GameBoy>();
       List<GameBoy> lank3 = new ArrayList<GameBoy>();
       List<GameBoy> lank4 = new ArrayList<GameBoy>();
       int secondChoice = 0;// 2차메뉴를 담을 변수

       String searchName = null;

       void checkFirstRank() {

           // 이 리스트가 속이 들은지 빈지 알려주는 메소드
           if (lank1.isEmpty()) {
               System.out.println("PLAY THE GAME FIRST...!");

           } else {

               // ALL RANK 또는 자가갱신 확인을 위해 나눈다
               while (true) {

                   secondChoice = choiceMenu2();
                   switch (secondChoice) {
                   case 1:
                       System.out.println("==GAME RANK==");
                       showFirstRank();
                       break;
                   case 2:

                       System.out.println("ENTER NAME");
                       searchName = GameBasic.key.nextLine();
                       int index = -1;
                       // 이름이 있는지 검색
                       Collections.sort(lank1, new Comprator());
                       for (int i = 0; i < lank1.size(); i++) {
                           if (searchName.equals(lank1.get(i).name)) {
                               index = i;
                               break;
                           }
                       }
                       if (index < 0) {
                           System.out.println("PlAY THE GAME FIRST..");
                       } else {

                           for (int i = 0; i < lank1.size(); i++) {
                               if (searchName.equals(lank1.get(i).name)) {
                                   System.out.println("-----" + (i + 1) + "등" + "-----");
                                   lank1.get(i).showData();// 수정
                                   break;
                               }
                           }

                       }
                       break;
                   case 3:
                       System.out.println("BACK");
                       break;
                   default:
                       System.out.println("머! 똑바로 숫자 안 누르나 ! 팍 쳐!");
                       break;
                   }
                   // 이전 화면으로 돌아가기
                   if (secondChoice == 3) {
                       break;
                   }

               }
           }
       }
   void checkSecondRank() {

           // 이 리스트가 속이 들은지 빈지 알려주는 메소드
           if (lank2.isEmpty()) {
               System.out.println("PLAY THE GAME FIRST...!");

           } else {

               // ALL RANK 또는 자가갱신 확인을 위해 나눈다
               while (true) {

                   secondChoice = choiceMenu2();
                   switch (secondChoice) {
                   case 1:
                       System.out.println("==GAME RANK==");
                       showSecondRank();
                       break;
                   case 2:

                       System.out.println("ENTER NAME");
                       searchName = GameBasic.key.nextLine();
                       int index = -1;
                       // 이름이 있는지 검색
                       Collections.sort(lank2, new Comprator());
                       for (int i = 0; i < lank2.size(); i++) {
                           if (searchName.equals(lank2.get(i).name)) {
                               index = i;
                               break;
                           }
                       }
                       if (index < 0) {
                           System.out.println("PlAY THE GAME FIRST..");
                       } else {

                           for (int i = 0; i < lank2.size(); i++) {
                               if (searchName.equals(lank2.get(i).name)) {
                                   System.out.println("-----" + (i + 1) + "등" + "-----");
                                   lank2.get(i).showData();// 수정
                                   break;
                               }
                           }

                       }
                       break;
                   case 3:
                       System.out.println("BACK");
                       break;
                   default:
                       System.out.println("머! 똑바로 숫자 안 누르나 ! 팍 쳐!");
                       break;
                   }
                   // 이전 화면으로 돌아가기
                   if (secondChoice == 3) {
                       break;
                   }

               }
           }
       }
   void checkThirdRank() {

       if (lank3.isEmpty()) {
           System.out.println("PLAY THE GAME FIRST...!");

       } else {

           // ALL RANK 또는 자가갱신 확인을 위해 나눈다
           while (true) {

               secondChoice = choiceMenu2();
               switch (secondChoice) {
               case 1:
                   System.out.println("==GAME RANK==");
                   showThirdRank();
                   break;
               case 2:

                   System.out.println("ENTER NAME");
                   searchName = GameBasic.key.nextLine();
                   int index = -1;
                   // 이름이 있는지 검색
                   Collections.sort(lank3, new Comprator());
                   for (int i = 0; i < lank3.size(); i++) {
                       if (searchName.equals(lank3.get(i).name)) {
                           index = i;
                           break;
                       }
                   }
                   if (index < 0) {
                       System.out.println("PlAY THE GAME FIRST..");
                   } else {

                       for (int i = 0; i < lank3.size(); i++) {
                           if (searchName.equals(lank3.get(i).name)) {
                               System.out.println("-----" + (i + 1) + "등" + "-----");
                               lank3.get(i).showData();// 수정
                               break;
                           }
                       }

                   }
                   break;
               case 3:
                   System.out.println("BACK");
                   break;
               default:
                   System.out.println("머! 똑바로 숫자 안 누르나 ! 팍 쳐!");
                   break;
               }
               // 이전 화면으로 돌아가기
               if (secondChoice == 3) {
                   break;
               }

           }
       }
   }
   void checkFourthRank() {

       // 이 리스트가 속이 들은지 빈지 알려주는 메소드
       if (lank4.isEmpty()) {
           System.out.println("PLAY THE GAME FIRST...!");

       } else {

           // ALL RANK 또는 자가갱신 확인을 위해 나눈다
           while (true) {

               secondChoice = choiceMenu2();
               switch (secondChoice) {
               case 1:
                   System.out.println("==GAME RANK==");
                   showFourthRank();
                   break;
               case 2:

                   System.out.println("ENTER NAME");
                   searchName = GameBasic.key.nextLine();
                   int index = -1;
                   // 이름이 있는지 검색
                   Collections.sort(lank4, new Comprator());
                   for (int i = 0; i < lank4.size(); i++) {
                       if (searchName.equals(lank4.get(i).name)) {
                           index = i;
                           break;
                       }
                   }
                   if (index < 0) {
                       System.out.println("PlAY THE GAME FIRST..");
                   } else {

                       for (int i = 0; i < lank4.size(); i++) {
                           if (searchName.equals(lank4.get(i).name)) {
                               System.out.println("-----" + (i + 1) + "등" + "-----");
                               lank4.get(i).showData();// 수정
                               break;
                           }
                       }

                   }
                   break;
               case 3:
                   System.out.println("BACK");
                   break;
               default:
                   System.out.println("머! 똑바로 숫자 안 누르나 ! 팍 쳐!");
                   break;
               }
               // 이전 화면으로 돌아가기
               if (secondChoice == 3) {
                   break;
               }

           }
       }
   }
   //2차 선택화면
   int choiceMenu2() {
       int choice = 0;
       System.out.println("1.ALL RANK\n2.MY RANK\n3.BACK");
       // ALL RANK 또는 자가갱신 확인을 위해 나눈다
       try {
           choice = GameBasic.key.nextInt();
       } catch (InputMismatchException e) {
       }
       GameBasic.key.nextLine();
       return choice;
   }
}
```



6. 게임랭킹관리

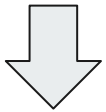
3. 코드의 반복 - 후

```
package lankTest;

import java.util.ArrayList;

public class LanlManger {
    private static LanlManger Lm;

    private LanlManger() {
    }

    public static LanlManger getInctance() {
        if (Lm == null) {
            Lm = new LanlManger();
        }
        return Lm;
    }

    List<GameBoy> lank1 = new ArrayList<GameBoy>();
    List<GameBoy> lank2 = new ArrayList<GameBoy>();
    List<GameBoy> lank3 = new ArrayList<GameBoy>();
    List<GameBoy> lank4 = new ArrayList<GameBoy>();
    int secondChoice = 0;// 2차메뉴를 담을 변수
    List<GameBoy> list;
    String searchName = null;

    void checkRank(List<GameBoy> list) {
        this.list = list;
        // 이 리스트가 죽이 들은지 빈지 알려주는 메소드
        if (list.isEmpty()) {
            System.out.println("PLAY THE GAME FIRST...!");

        } else {

            // ALL RANK 또는 자가랭킹 확인을 위해 나눈다
            while (true) {

                secondChoice = choiceMenu2();
                switch (secondChoice) {
                case 1:
                    System.out.println("==GAME RANK==");
                    showAllRank();
                    break;
                case 2:

                    System.out.println("ENTER NAME");
                    searchName = GameBasic.key.nextLine();
                    int index = -1;
                    // 이름이 있는지 검색
                    Collections.sort(list, new Comprator());
                    for (int i = 0; i < list.size(); i++) {
                        if (searchName.equals(list.get(i).name)) {
                            index = i;
                            break;
                        }
                    }
                    if (index < 0) {
                        System.out.println("PLAY THE GAME FIRST..");
                    } else {

                        for (int i = 0; i < list.size(); i++) {
                            if (searchName.equals(list.get(i).name)) {
                                System.out.println("-----" + (i + 1) + "등" + "-----");
                                list.get(i).showData();// 수정
                                break;
                            }
                        }

                    }
                    break;
                case 3:
                    return;
                }
            }
        }
    }

    // 전체 랭킹 출력
    void showAllRank() {

        Collections.sort(list, new Comprator());
        for (int i = 0; i < list.size(); i++) {
            System.out.println("-----" + (i + 1) + "등" + "-----");
            list.get(i).showData();
        }
        return;
    }

//2차 선택화면
    int choiceMenu2() {
        int choice = 0;
        System.out.println("1.ALL RANK\n2.MY RANK\n3.BACK");
        // ALL RANK 또는 자가랭킹 확인을 위해 나눈다
        try {
            choice = GameBasic.key.nextInt();
        } catch (InputMismatchException e) {
        }
        GameBasic.key.nextLine();
        return choice;
    }
}
```



7. 협업 과정에서의 문제점

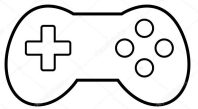
같은 기능을 하는 변수들이 다양한 이름으로 정의되어 있어서 합치는 과정에서 난항을 겪음



인터페이스, 추상클래스 이용의 중요성...!

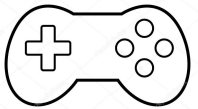
팀원들간의 코드 리뷰 과정이 다소 아쉬웠음

지속적인 코드리뷰를 통한 각 팀원들의 전체 구조 및 코드 이해 향상



8. 프로젝트의 개선방향

- **공통**
  - 효율적인 전체 구조 및 흐름에 대한 고민
  - 중복되는 코드 정리
- **메인/회원관리**
  - 로그인된 유저를 유지시킨채로 프로그램 진행이 더 용이 할 수 있도록 개선해야한다.
  - 비밀번호 암호화가 필요하다.
  - 회원가입과 로그인에 유효성검사가 더 필요하다.(ex) 비밀번호는 영문자,숫자섞어서 가능하도록)
- **게임1(억울한 사형수를 구해라!)**
  - 랜덤 정답 단어 뽑는 과정에서 외부 소스를 통해 더 많은 데이터 읽어오기
  - 정답단어와 빈칸의 대조: char[] 리스트로 받지 않고 string 관련 매서드 이용하는 법
  - 힌트주기 부분 개선



8. 프로젝트의 개선방향

- **게임2(영화제목을 맞춰라!)**
  - 문제 추가 후 랜덤하게 문제가 등장하게 수정
  - JOptionpane 을 사용하여 정답 입력 팝업창 생성
  - 중복되는 코드들의 정리